Avdeling for informatikk og e-læring, Høgskolen i Sør-Trøndelag

Anette Wrålsen 2011/2012

Lærestoffet er utviklet for faget LC372D Publisering på internett

Introduksjon til CSS

I denne leksjonen vil du få en introduksjon til CSS – du får vite litt om om hva CSS er, om hvorfor vi bruker det og ikke minst om hvordan vi bruker det.

Hva er CSS?

CSS er et språk som brukes for å bestemme *presentasjonen* av nettsider. Det betyr at CSS lar oss styre farger, fonter, bakgrunner og i stor grad også layouten til en nettside.

Da World Wide Web begynte å slå gjennom på starten nittitallet ble det fler og fler som hadde hjemmesider, og mange ønsket å kunne style siden utover det som enkel HTML tilbød. Dette førte til at det ble utviklet HTML-tagger og -attributter som kunne brukes til styling.

Utviklingen av ekstra HTML-tagger og -attributter for styling førte imidlertid til noen problemer. Et problem var at forskjellige nettlesere begynte å utvikle *egne* tagger, dvs. tagger som kun fungerte i den spesifikke nettleseren. Det betød at utviklere måtte lage egne versjoner av nettsidene sine for forskjellige nettlesere for å sikre seg at de så like ut for alle brukere. I tillegg kunne det variere fra nettleser til nettleser hva som var mulig å få til. Dette var selvsagt et konkurransefortrinn for de enkelte nettleserne, men det var samtidig et stort problem for utviklere som måtte lage og vedlikeholde kode som skulle gi best mulig resultat i flest mulig nettlesere.

Et annet problem var at det rett og slett ga *mye* og *uoversiktlig* kode. Innhold og presentasjon ble sauset sammen, og mange begynte å bruke de originale HTML-taggene på måter som de slettes ikke var laget for. For eksempel var det lenge vanlig å bruke tabeller for layout. Dette fungerer tålelig bra, men er langt fra optimalt og slettes ikke det tabeller var tenkt for. Tabeller var tenkt for å strukturere informasjon, ikke for å angi layout. I tillegg sluttet mange å bruke tagger som for eksempel headertaggene (`<h1>`, `<h2>` osv.) og brukte heller `<font>`-taggen - en egen tagg som ble laget spesifikt for å kunne style tekst. Det gjorde også sidene vanskeligere å lese med nettlesere tilpasset brukere med spesielle behov, siden disse trenger å forstå strukturen til en nettside for kunne presentere den effektivt.

CSS, eller stilark som de ofte kalles på norsk, ble utviklet av Håkon Wium Lie (som for tiden er teknologidirektør i Opera Software) og Bert Bos på midten av nittitallet. Stilark ble imidlertid ikke implementert fullt ut i noen nettleser før i 2000 og har først blitt vanlig å bruke de siste årene. De er altså egne kodesnutter som du kan assosiere

med en HTML-fil, og som lar deg spesifisere presentasjonen av HTML-taggene i denne filen. Det gir oss mer oversiktlig kode, og det gir oss bedre måter å kode layout på. I tillegg så er CSS en del av anbefalingene til W3C - World Wide Web Consortium, som jobber for å ha felles nettstandarder for Verdensveven, slik at vi unngår problemene nevnt over med mange versjoner av nettsider tilpasset forskjellige nettlesere. Man har i stor grad lyktes med denne strategien, og nettleserne kappes nå stort sett om å implementere standarder og nye versjoner av hhv. (X)HTML og CSS fortest mulig i stedet for å lage egne versjoner.

> For å teste om en nettleser forholder seg til de gjeldene standardene kan du utføre en *syretest* på den: Gå til www.acidtests.org, sørg for at nettleseren er satt til standardinnstillingene og ta den nyeste testen. Disse testene er laget av Web Standards Project (en organisasjon som jobber for å fremme bruken av nettstandardene til W3C). Test nummer tre, Acid3, fokuserer mest på CSS men tester også andre ting. Den fikk stor påvirkning på utviklingen av mange nettlesere på den måten at det gikk kort tid fra testen kom ut til de fleste store nettleserne sørget for at de scoret høyt. Les mer om den for eksempel på Wikipedia.

Merk også at XHTML og HTML5 begge ikke tillater taggen `<font>` og andre tagger som ble laget kun for å style nettsider – de krever begge at du bruker CSS til styling.

I likhet med (X)HTML så kommer CSS i versjoner. Ennå er CSS2 den anbefalte standarden fra W3C, men de fleste nettlesere implementerer også CSS 2.1 og denne versjonen regnes som rimelig standard. CSS3 er under utvikling, og noen nettlesere har også implementert elementer fra denne. Du bør imidlertid alltid teste koden i de vanligste nettleserne når du bruker ferske elementer. Selvsagt bør du alltid teste koden din i de vanligste nettleserne uansett, men det blir ekstra viktig i dette tilfellet. I tillegg kan det være greitt å være klar over at ikke alle nettlesere implementerer CSS helt likt. Forskjellene er imidlertid som regel små – men er du veldig nøye på hvordan siden din skal se ut, er dette noe å være obs på.

Hvorfor CSS?

Men hvorfor lære seg enda et språk når HTML i prinsippet kan gjøre samme jobben? En grunn har vi allerede vært inne på – det gjør koden uoversiktlig og mer utsatt for feil. I tillegg er stilark både kraftigere og mer fleksibelt enn HTML. For eksempel kan du koble samme stilark til mange (X)HTML-sider, og slik spare deg masse koding og samtidig spare brukeren for nedlasting. Det forenkler også veldig jobben med å endre et design – alt du trenger å gjøre er å endre en kommando i en CSS-fil for å endre fonten på alle `<h1>`-headere, i stedet for å gå inn i alle dokumenter og endre alle forekomster av denne taggen. Du kan også koble flere stilark til samme (X)HTML-side, for eksempel for å kunne tilby forskjellig utseende og layout alt etter som leseren surfer på en datamaskin eller en mobiltelefon. For komplekse nettsteder kan det også være praktisk å ha forskjellige deler av layouten i forskjellige filer.

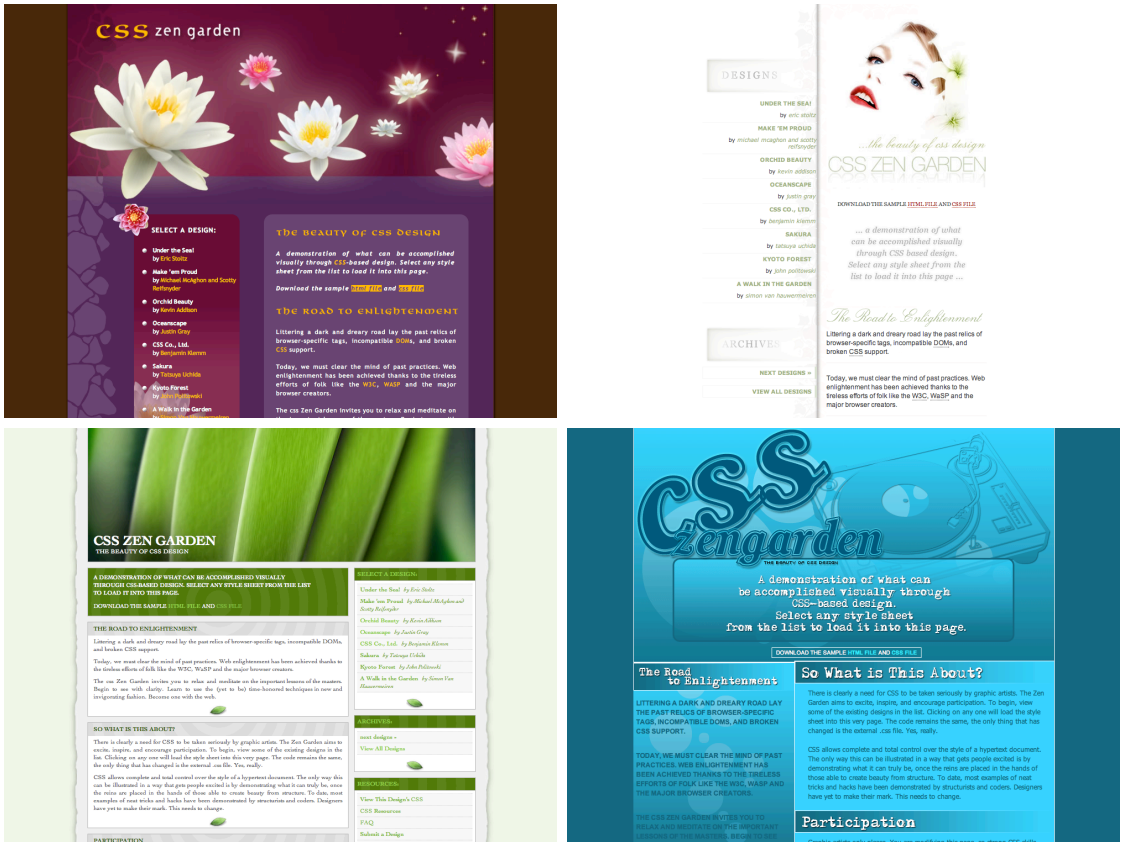
I tillegg kan brukerne ha egne stilark lokalt på nettleseren som overstyrer dine. Dette gjør at svaksynte kan endre skriftstørrelser og kontraster, og fargeblinde kan endre fargene.

Generelt sier vi at å bruke både HTML og CSS gir *separasjon mellom struktur og presentasjon*. HTML gir strukturen til siden mens stilarkene beskriver presentasjonen. I tillegg er CSS kraftigere enn HTML, og gir flere muligheter. For eksempel kan du ha forskjellige bakgrunnsbilder i forskjellige regioner i bildet. Dette tillater ikke HTML på noen god måte.

CSS Zen Garden

CSS Zen Garden er et nettsted som demonstrerer kraften i CSS veldig godt. Dette nettstedet ble laget nettopp for å demonstrere for webutviklere at CSS kan gjøre alt som HTML kan, og mer til. Alle sidene på dette nettstedet er akkurat samme HTML-fil med forskjellige stilark knyttet til seg!

Eksempel - noen design fra CSS Zen Garden



Hvordan bruker vi CSS?

Da begynner vi å bli klar for å finne ut hvordan vi faktisk koder CSS. Før vi fortsetter ønsker jeg å bemerke at dette er en introduksjon til hvordan vi bruker CSS og til viktige prinsipper som ligger til grunn for hvordan CSS bestemmer layout, basert av eksempler. Nettet er fullt av gode tutorials og oppslagsverk på dette området, og det meste du kommer til å lure på om CSS har noen lurt på før deg, slik at et godt formulert nettsøk høyst sannsynlig vil kunne hjelpe deg med å løse problemet.

Syntaks

Den grunnleggende CSS-syntaksen er enkel:

```
selektor {attributt: verdi;}
```

Selektoren forteller hvilken HTML-tagg du ønsker å style, attributten forteller hvilket aspekt ved taggen du vil endre, og verdien forteller hva du vil endre den til. Et konkret eksempel følger:

```
p {color: pink;}
```

Denne koden gjøre at alle avsnitt (dvs. alle deler av HTML-dokumentet ditt som er tagget med `<p>`-taggen) får rosa tekst. Vi kan også være litt effektive og skrive flere formateringer mellom samme klammeparenteser slik:

```
p, h1 {color: pink;
       background: yellow;}
```

Denne regelen gjør at alle avsnitt og `<h1>`-overskrifter får rosa tekst og gul bakgrunn og gir følgende vakre resultat på en meget enkel eksempelnettside:

Her er noen eksempler på CSS-syntaks som gir deg enda større fleksibilitet:

```
p.sammendrag {color: green;}
```

Denne kodesnutten gir grønn tekst til alle avsnitt som vi har gitt HTML-attributten `class="sammendrag"`. Klasser er en måte å gruppere tagger som du vil gi lik styling på. Koden

```
#advarsel {color: red;}
```

referer spesifikt til et HTML-element som har `id="advarsel"` som attributt. Forskjellen på `class` og `id` er at `class` kan brukes på flere elementer, mens `id` identifiserer et element unikt (og kan dermed bare brukes på et element i hvert dokument). Og

```
p em {color: white;}
```

gjør at all tekst inneholdt i et avsnitt som igjen er merket med en `<em>`-tagg (utheving) vil få hvit farge. Dette vil altså ikke påvirke uthevet tekst som *ikke* er i et avsnitt. CSS har også et jokertegn, det vil si et tegn som betyr "alle". Skriver du

```
* {color: white;}
```

blir all tekst i dokumentet ditt hvit.

Kommentarer

All kode bør kommenteres, og CSS er inget unntak. Det betyr at du bør ha med kommentarer i koden som forklarer hva du ønsker at koden skal gjøre. Syntaksen for CSS-kommentarer er som følger:

```
/* Dette er en kommentar. */
```

Hvordan kobler vi et stilark til en (X)HTML-fil?

Det er tre måter å knytte CSS-kode til en (X)HTML-fil på: inline, som internt stilark og som eksternt stilark.

Inline styling

Du kan legge CSS-kode som en attributt i en (X)HTML-tagg. Dette er en måte som egentlig ikke er veldig god siden den fjerner mange av fordelene med CSS, men kan kanskje være nyttig i spesialtilfeller der du bare trenger å style en enkelt tagg. Syntaks:

```
<p style="color: green;"> Dette er et grønt avsnitt. </p>
```

Internt stilark

Interne stilark er egne kodesnutter du legger i hodedelen til et (X)HTML-dokument. De bør først og fremst brukes når du kun skal style et enkeltdokument. Dette bevarer separasjonen mellom presentasjon og struktur til en nettside, men mangler mange av effektiviseringsfordelene ved CSS siden koden bare er tilgjengelig for (X)HTML-dokumentet det står i. Syntaks:

```
<head>
<style type="text/css">
p, h1 {color: pink;
       background: yellow;}
</style>
</head>
```

Eksternt stilark

Her kommer endelig CSS til sin fulle rett. Denne metoden går ut på å ha et eget stilark (en ren tekstfil med filending .css) som knyttes til (X)HTML-dokumentet. Du kan knytte så mange stilark du vil til samme (X)HTML-dokument, og du kan til og med bruke andres stilark (så lenge stilarkene er tilgjengelige på nett og du kjenner nettadressen). Stien til stilarket ligger i hodedelen av dokumentet, og syntaksen er som følger:

```
<head>
<link rel="stylesheet" type="text/css" href="stilark.css" />
</head>
```

Denne koden gjør at (X)HTML-dokumentet styles med koden i filen stilark.css. Du kjenner kanskje igjen `href`-attributten som brukes i lenketagger? Den fungerer akkurat likt – det vil si at du må angi enten en relativ sti til det aktuelle stilarket, eller en fullstendig nettadresse. Ligger det i samme mappe som (X)HTML-dokumentet, kan du bare skrive filnavnet slik som i eksemplet over.

Å style med CSS

Vi skal nå gå litt mer i dybden på hvordan CSS fungerer.

Enkel styling - farger og bakgrunner

Vi har allerede sett noen enkle stilregler. Her er litt mer om noen typer attributter:

Farger

Farger kan angis enten slik som vi har sett i eksemplene tidligere, nemlig som en av 17 forhåndsdefinerte farger (CSS 2.1) som `red`, `green`, `olive`, `lime` og `blue`. I tillegg aksepterer de fleste nettlesere enda flere fargenavn enn de 17 forhåndsdefinerte. Men hvis du vil ha full kontroll over hvilken farge som vises bør du uansett angi RGB-koden. Da må du angi hvor mye rødt, grønt og blått som fargen inneholder og det kan du gjøre på en av følgende måter:

```
/* angir i prosent: */
h2 {color: rgb(0%, 0%, 100%);}

/* angir på skala fra 0 til 255: */
h2 {color: rgb(0, 0, 255);}

/* angir heksadesimalt: */
h2 {color: #0000FF;}
```

```
/* angir på heksadesimal kortform: */
h2 {color: #00F;}
```

Alle disse kodelinjene gjør nøyaktig det samme – de farger tekst i `<h2>`-tagger blå. Bruk metoden du foretrekker selv!

Lengdeenheter

På samme måte som CSS tillater mange måter å angi farger på, kan man bruke mange lengdeenheter. Både tommer (`in`), centimeter (`cm`), millimeter (`mm`) og typografiske enheter som punkter (`pt`) og picas (`pc`) kan brukes. Du kan også bruke relative enheter som ems (`em`). En em er størrelsen du allerede har gitt `font-size`-attributten til dette elementet. Det vil si at du kan definere størrelsen på andre attributter relativt til denne ved for eksempel å definere den til å være 1.2 ganger så stor.

```
h2 {font-size: 14px;
    margin-left: 1.2em;}
```

Bakgrunner

Et av de store layoutproblemene med HTML var at det var vanskelig å la forskjellige deler av siden ha forskjellige bakgrunner. Dette er ikke noe problem med CSS – du kan rett og slett bruke attributtene `background-color` og `background-image` for å bestemme bakgrunnen til enkeltelementer. Her er et eksempel:

```
h1 {background-color: aqua;}
p  {background-image: url(http://www.eksempel.no/bilde.jpg);}
```

Der fikk vi illustrert en ting til i samme slengen – nemlig hvordan man angir en URL i CSS! Du kan selvsagt i stedet angi en sti til et bilde på nettstedet ditt.

I teorien kan du angi bakgrunner til alle typer elementer på denne måten, men det kan variere litt fra nettleser til nettleser hvordan dette håndteres

Skrifttyper

Når vi skal velge skrifttyper (fonter) i CSS, gjøres dette i ved hjelp av attributter som `font-family`, `font-size` og `font-style`. `Font-family` brukes for å angi selve skrifttypen:

```
p  {font-family: Arial, Helvetica, sans-serif}
```

Vi bør alltid liste opp flere alternativer, og avslutte med en generisk *font-familie* som serif, sans-serif eller monospace. Dette er fordi vi er avhengige av at brukeren har skrifttypen vi ønsker å bruke installert på maskinen sin (dette er en sannhet med modifikasjoner, se avsnittet om CSS3 til slutt). Når vi lister opp skrifttyper på denne måter vil nettleseren først sjekke om brukeren har den første skrifttypen på lista, hvis ikke denne er tilgjengelig går den videre til nummer to, og slik fortsetter den til den finner noe den har. I verste fall har den ingen av dem, og må velge noe selv. Da er det

bra å ha med en font-familie til slutt, for da er du i det minste garantert at brukeren får presentert teksten med en skrifttype som ligner på den du ønsker.

`Font-size` spesifiserer størrelsen på tekst, og den må angis med en av de godkjente enhetene (se avsnittet om lengdeenheter, der er et eksempel på bruk av denne attributten). `Font-style`-attributten tar verdiene `normal`, `italic` (kursiv) og `oblique` (en variant av kursiv).

```
p.kursiv {font-style: italic;}
```

Styling av tekst

Når vi styler tekst, er det flere relevant attributter. De viktigste er `text-align` (styrer justering av teksten og tar verdiene `center`, `left`, `right` og `justify`) og `text-decoration` (lar deg bestemme ting som gjennomstreking, understreking, overstreking og blinking med verdiene `overline`, `line-through`, `underline`, `blink` og `none`). Denne brukes ofte til å fjerne streker under lenker ved å sette

```
a {text-decoration: none;}
```

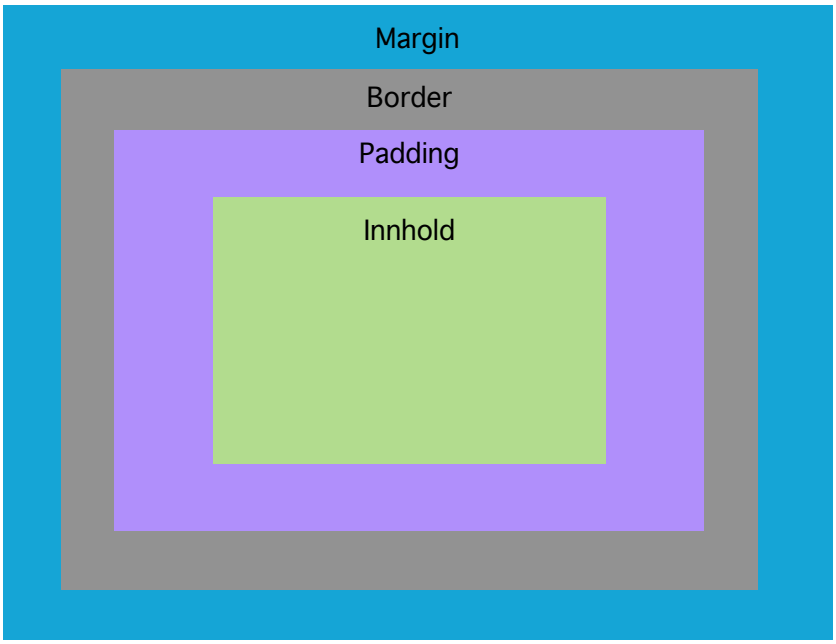
I tillegg har vi attributten `text-indent`, som lar deg bestemme innrykk på første linje av teksten:

```
p {text-indent: 2em;}
```

Boksmodellen

CSS har som utgangspunkt at alle (X)HTML-elementer er inneholdt i en *boks* på skjermen. De såkalt *blokkelementene* genererer et helt eget rektangel på skjermen, og bryter flyten i dokumentet. Det betyr i praksis at det er linjeskift før og etter dem. Den andre hovedtypen elementer, *inlineelementer*, bryter ikke flyten. Eksemper på blokkelementer er `<p>`-elementer, `<div>`-elementer og de forskjellige overskriftene (headingene). Eksempler på inlineelementer er `<span>` og `<em>`. Merk at det er mulig å overstyre om et element skal vises inline eller som en blokk.

Når vi skal bestemme hvordan en slik boks ser ut, må vi ta hensyn til *boksmodellen* i CSS. Boksmodellen går ut på at alle bokser har fire typer områder, nemlig margin, border, padding og innhold. Disse områdene illustreres best med en figur:



Margin angir luften rundt elementet, border en eventuell ramme rundt, paddingen er luften mellom elementet og rammen mens innholdet er, vel, innholdet. Alle disse kans stilles inn som du vil. Du kan også stille inn `height` og `width`. Merk at dette er høyden og bredden på *innholdet*, og ikke på hele boksen. For finne høyden (eller bredden) til hele boksen må du legge til padding, borderen og margin. Hva tror du følgende kode gjør?

```
p {background: red;
   height: 14%;
   padding: 15px;
   border: 10px;
   border-style: solid;
   border-color: green;
   margin: 30px;}
```

CSS og layout: -taggen, floating og positioning

En tagg som er viktig når det gjelder CSS og og layout, er `<div>`-taggen. Det er en forkortelse for *division*, og er rett og slett en tagg som angir en *inndeling* av nettsiden. Ved å plassere innholdet i forskjellige slike tagger kan vi bruke CSS til å styre hvordan de ser ut og plasseres på skjermen. Dette er en av de vanligste måtene å lage layout med CSS på. HTML5 kommer også med egne tagger for navigasjonselementer som for eksempel menyer (`<nav>`) og footere (`<footer>`) nettopp fordi div'er med disse funksjonene er såpass vanlige at de nærmest er standard i nettsider nå. En meget enkel nettside inndelt ved hjelp av div'er kan ha følgende kode:

```
<head>
<style type="text/css">
   #logo {background: red;}
   #informasjon {background: green;}
```

```
      #salg {background: blue;}
    </style>
    </head>

    <body>
    <div id="logo">
      Nettbutikken Stort&Lite
    </div>
    <div id="informasjon">
      Nettbutikken Stort&Lite er en nettbutikk som selger både
    stort og smått.
    </div>
    <div id="salg">
      I dag har vi salg på smått.
    </div>
    </body>
```

Her er det tre div'er, og de har hver sin unike id som lar oss style dem spesifikt med hver sin bakgrunnsfarge.

`<div>`-elementet har attributten `position` som du kan bruke til å styre hvor på nettsiden elementet vises. Her er noen verdier den kan ta og hvordan de virker:

- ✓ `static` – defaultverdien. Det betyr at elementet kommer i den rekkefølgen det står i HTML-dokumentet, den såkalte *normale posisjonen*. Da blir det linjeskift før og etter elementet.
- ✓ `fixed` – her står elementet på en fast plass i forhold til nettleservinduet, og flytter seg ikke selv om det scrolles. Det tas ut av den normale flyten i dokumentet. Det kan også overlappe med andre elementer.
- ✓ `relative` – her står elementet i utgangspunktet i sin naturlige flyt, men du kan flytte ut til en ny posisjon *relativt* til den normale posisjonen (for eksempel litt før, litt over eller litt under).

For å lage virkelig kompleks layout må man ofte legge `div`-elementer inni `div`-elementer – såkalt *nesting.*

En annen viktig attributt er `float`. Floating er å la andre elementer kan «flyte» rundt et element, slik som denne teksten flyter rundt det grønne rektanglet til venstre. Flyting flytter dermed elementet til en viss grad ut av den naturlige flyten til dokumentet. Uten muligheten til å gjøre dette, ville layoutmulighetene til CSS vært sterkt begrensede. Det gir deg muligheten til å plassere elementer som du vil. Merk at det kan være ganske krevende å få avansert layout *akkurat* slik du ønsker. Under er en oversikt over de viktigste verdiene til denne attributten:

- ✓ `left` – dette flytter et element helt ut til venstre i nettleservinduet, og lar de andre elementene flyte rundt det.
- ✓ `right` – tilsvarende flytter dette et element helt til høyre i nettleservinduet.
- ✓ `none` – dette er defaultverdien til attributten.

(Hvis du lurer på hvorfor det er mulig å angi verdier som jo er defaultene uansett, kan dette for eksempel være nyttig hvis du ønsker å overstyre verdier gitt i andre stilark.)

Før vi går videre vil jeg spesifisere at det ikke bare er `<div>`-elementer som har attributtene `position` og `float`, du kan bruke dem også på andre (X)HTML-tagger (som for eksempel bilder). Det kan være litt vanskelig å forstå akkurat hva forskjellige kombinasjoner av verdier for `position` og `float` gjør med flyten i et dokument, så jeg anbefaler å bruke litt tid på å eksperimentere for å bli kjent med disse attributtene.

Spesifisitet

Noen ganger når vi har mye CSS-kode knyttet til en nettside, får vi mange motstridende CSS-regler som styrer samme element. For eksempel kan vi ha koden

```
h2 {color: blue;}
```

i et eksternt stilark som gjelder for hele nettstedet, koden

```
.dikttittel {color: green;}
```

i et eksternt stilark som gjelder bare diktene på nettstedet, og

```
#mauren {color: red;}
```

i hodedelen av dokumentet som blant annet inneholder diktet «Mauren». Hvordan i all verden blir da elementet

```
<h2 class="dikttittel" id="mauren" >Mauren</h2>
```

seende ut? Svaret ligger i *spesifisitet.* Stilark med motstridende regler prioriteres etter klart definerte regler. Disse reglene baserer seg på at man regner ut en verdi knyttet til hver regel kalt spesifisiteten til regelen, og hvis to regler kommer i konflikt vinner den med høyest spesifisitet. Vi skal ikke gå inn på nøyaktig hvordan dette gjøres, men som hovedregel kan vi si at jo mer spesifikt en regel peker ut et element (derav navnet spesifisitet), jo høyere prioritet får regelen. Hvis to regler har samme prioritet, er det rett og slett den siste som gjelder. Kode i hodedelen til et dokument vinner også over kode i et eksternt stilark.

Det betyr at i vårt tilfelle vil dikttittelen «Mauren» bli rød – fordi det er mer spesifikt å angi et element ved hjelp av id i hodedelen av et dokument enn ved klasse eller tag i eksterne stilark. Hadde klasse-regelen og tag-regelen konkurrert, ville klasseregelen vunnet. Hadde to klasse-regler fra samme stilark konkurrert, ville den *siste* vunnet.

Her er en artikkel som forklarer mer om CSS og spesifisitet for dem som vil vite mer:
http://www.smashingmagazine.com/2007/07/27/css-specificity-things-you-should-know/
Her er en spesifisitetskalkulator der du kan skrive inn kode og få ut spesifisiteten:
http://www.suzyit.com/tools/specificity.php

Pseudoklasser og pseudoelementer

Pseudoklasser og pseudoelementer er klasser og elementer som ikke egentlig finnes i dokumentet ditt, men som velges ut etter "omstendighetene". For eksempel er elementet musepekeren beveger seg over akkurat nå en mulig pseudoklasse. Syntaks for regler for pseudoklasser kan du se i eksemplene under. Vi kan for eksempel bruke pseudoklasser til å skifte farge på lenker:

```
a:link {color:blue;}      /* ubesøkt lenke */
a:visited {color: red;}   /* besøkt lenke */
a:hover {color: green;}   /* lenke musa beveger seg over */
a:active {color:yellow;}  /* aktiv lenke */
```

Merk at disse stilreglene bør gis i rekkefølgen vist over. Grunnen til dette er at reglene kan komme i konflikt (for eksempel kan en lenke både være besøkt og ha musa over seg samtidig), og i så fall velger nettleseren å bruke den regelen som kommer sist (dette er et eksempel på det vi kaller *spesifisitet* i CSS). Det betyr at hvis du setter linje to etter linje tre, vil besøkte lenker bli røde og ikke grønne selv når musepekeren er over dem.

Pseudoelementer er en måte å referere til den første bokstaven eller den første linja i et element, eller til elementet før eller etter et gitt element på. Her er et eksempel:

```
p:first-letter {color: green;}
```

Denne koden gjør første bokstav i alle avsnitt (`<p>`-tagger) grønne.

Merk at det kan variere litt fra nettleser til nettleser hvordan og i hvilken grad denne typen dynamisk design støttes.

Litt om CSS3

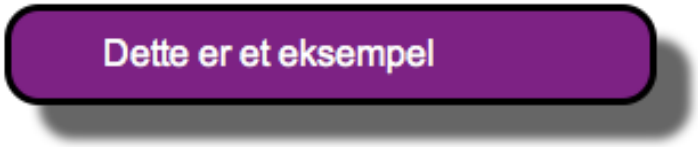
CSS3 er som tidligere nevnt en standard som fortsatt er under utvikling, men mange deler av denne standarden er allerede implementert i de fleste nettlesere (ofte er det bare Internet Explorer som er unntaket). Det vil si – den er stort sett implementer i *nyeste versjon* av de fleste nettlesere. CSS3 tilbyr mange avanserte former for styling, som å gi elementer og tekst skygge, å tilby en font på nettstedet (i stedet for at brukeren må ha fonten installert på maskinen sin), bedre kontroll over bakgrunnsbilder og enkle 2D- og 3D-transformasjoner samt animasjoner. Her er noen eksempler:

Border-egenskaper

Attributtene `border-radius` og `border-shadow` støttes av de fleste nettlesere nå. `Border-radius` lar deg lage avrundete kanter og `border-shadow` lar et bokselement kaste skygge, og begge deler er mye brukte effekter. Hittil har det stort sett blitt løst med drøssevis med små bildefiler og kompliserte (og gjerne lite fleksible layouter). Her er et eksempel:

```
div {border: 3px solid black;
     border-radius: 15px;
     box-shadow: 15px 15px 5px #666666;}
```

Denne koden (sammen med litt fargestyling) gir følgende boks:



Verdien til `border-radius`-attributten angir radiusen på sirkelen som gir de avrundede hjørnene (større radius betyr mer avrundet, prøv selv). De fire verdiene til `box-shadow` angir hhv. hvor langt under boksen skyggen skal gå, hvor langt til høyre for boksen skyggen skal gå, hvor skarpe kantene til skyggen skal være (større tall gir mer uskarpe kanter) og fargen på skyggen.

Det finnes også en attributt som lar deg bruke et bilde for å lage kanten rundt et bokselement, nemlig `border-image`. Denne er ikke like godt støttet som de to andre egenskapene.

CSS3 og skrifttyper: @font-face

CSS3 lar deg legge ved en skrifttype i en fil på nettstedet, slik at brukere ikke lenger trenger å ha installert skrifttypene du bruker på nettstedet. For å få til dette må du ha skrifttypen liggende på nettstedet i ttf-format (eller eot-format for IE). Deretter kan du definere et navn på skrifttypen, og senere referere til den på akkurat samme måter som andre skrifttyper (se avsnittet om CSS og fonter tidligere i leksjonen). Du gir navn til en skrifttype slik:

```
@font-face {font-family: minSkrifttype;
            src: url(navnPåSkrifttype.ttf),
                 url(navnPåSkrifttype.eot) format("opentype");}
```

Merk at den siste url'en der er til ære for Internet Explorer. Nå kan vi referere til skrifttypen som følger:

```
p  {font-family: minSkrifttype}
```

Videre lesning

Du har nå fått en introduksjon til mange av de viktigste sidene ved CSS. Denne gjennomgangen var alt annet enn uttømmende, og det kan skrives (og har blitt skrevet) tykke bøker om hvordan man koder avansert layout med CSS. CSS er også, i likhet med mange andre netteknologier, er i stadig endring og utvikling. Her er noen steder på nett der du kan lese og lære mer om stilark:

- ✓ http://www.w3schools.com/css/
- ✓ http://htmlhelp.com/reference/css/
- ✓ http://jigsaw.w3.org/css-validator/ – sjekk at koden din er korrekt her.
- ✓ http://people.opera.com/howcome/2006/phd/ – doktorgradsavhandlingen til Håkon Wium Lie om CSS.
- ✓ http://www.w3.org/Style/CSS/ – W3C om CSS-standarder.

Referanser

Meyer, Eric. 2007.
CSS: The Definitive Guide. Tredje utgave.
Sebastopol, CA, USA: O'Reilly Media.
ISBN 978-0-596-52733-4.

W3Schools, w3schools.com

Wikipedia, http://www.wikipedia.org.